Simulating Decorative Mosaics

Alejo Hausner

University of Toronto

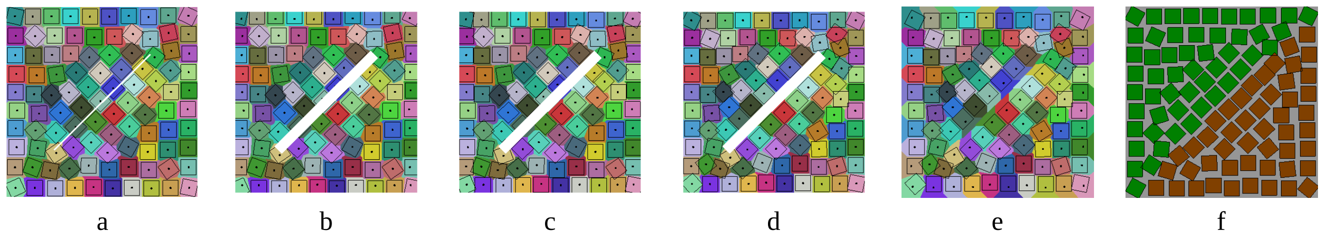


Figure 1: By overwriting voronoi regions, tile centroids are displaced away from an edge. Recentering tiles at their new centroids eventually moves them clear of the edge.

Abstract

This paper presents a method for simulating decorative tile mosaics. Such mosaics are challenging because the square tiles that comprise them must be packed tightly and yet must follow orientations chosen by the artist. Based on an existing image and user-selected edge features, the method can both reproduce the image's colours and emphasize the selected edges by placing tiles that follow the edges. The method uses centroidal voronoi diagrams which normally arrange points in regular hexagonal grids. By measuring distances with an manhattan metric whose main axis is adjusted locally to follow the chosen direction field, the centroidal diagram can be adapted to place tiles in curving square grids instead. Computing the centroidal voronoi diagram is made possible by leveraging the z-buffer algorithm available in many graphics cards.

1 Introduction

Artists often invent techniques later used in computer graphics. Tile mosaics, for example, are images made by cementing together small polygonal coloured patches. They are early examples of image synthesis techniques such as point sampling and rasters of pixels. However, ancient mosaicists avoided lining up their tiles in rectangular grids of pixels, knowing that to the eye such grids emphasize horizontal and vertical lines. These lines are an artifact which distracts the eye from seeing the image that the tiles describe. On the contrary, they took pains to align tiles to emphasize edges in the image at places of their own choosing. In Fig. 3, the artist carefully chose tile shapes and orientations to portray human faces.

We can understand their motivation by considering the information content of the resulting image. Aside from eliminating artifacts, a mosaic made up of N tiles can convey more information than an image made up of N pixels, simply because tiles carry extra information, such as position, shape, and orientation, which the human visual system can draw on.

1.1 The Problem

The goal of aligning square tiles with varying orientations is at odds with the goal of minimizing visible grout (the substrate that shows at gaps between tiles), or equivalently maximizing the area covered by coloured tiles. The two goals are opposed because square tiles can be arranged with maximal coverage only if all tiles have the same orientation. Our problem may be stated more formally as follows:

Formal Statement: Given a rectangular region I^2 in the plane R^2, and a vector field $\phi(x, y)$ defined on that region, find N sites $P_i(x_i, y_i)$ in I^2 and place N squares of side s, one at each P_i, oriented with sides approximately parallel to $\phi(x_i, y_i)$, such that all squares are disjoint and the area they cover is maximized.

The rectangular region covers a coloured image, and each square will be uniformly coloured, representing the part of the image it covers. The direction field ϕ can be general, but in our application tends to align tiles with edge features chosen by the user.

It may seem restrictive to use only square tiles, but we do so to simplify the problem to one of placing point particles. Hand-crafted mosaics tend to use mostly square tiles. Where they do not, we hope that distorting some tiles will approximate the artist's approach.

This paper presents a new technique for placing tiles. The method is general, and can be extended to problems in computer graphics and visualization, such as generating low-discrepancy sampling patterns and vector field visualizations. Mosaics are but one application.

The main contribution of this paper is a new method for finding low-energy configurations of particles. The novelty lies in the use of graphics hardware to automatically restrict the search for neighbouring particles. The method can be adapted to other inter-particle energy functions.

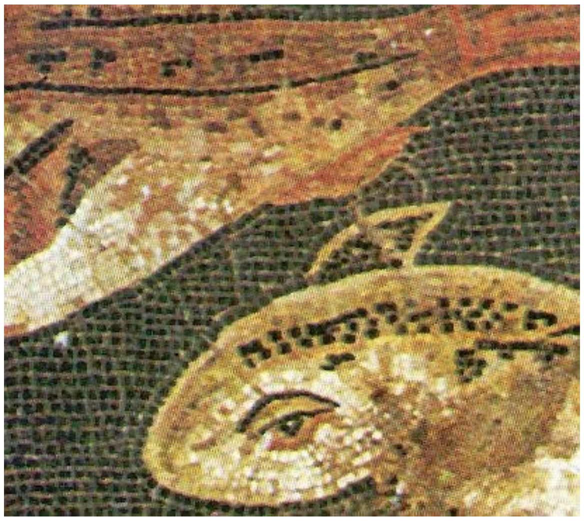


Figure 2: Detail from “Sea Creatures”, Museo Nazionale, Naples, 1st century BC.

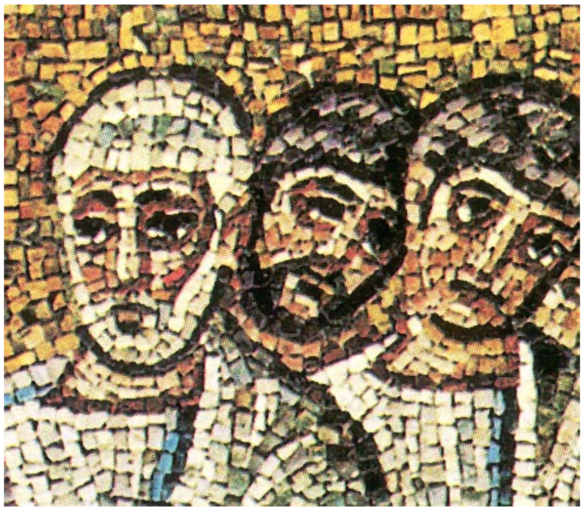


Figure 3: Detail from “The Betrayal”, St. Apollinarie Nuovo, Ravenna, early 6th century.

2 Related Work

Mosaic pictures survive from Greek and Roman times, 2000 years ago (Figures 2 and 3 give two examples). However, attempts to reproduce them using software are recent. Adobe Photoshop© provides a mosaic-effect plugin with square tiles, but this merely reduces spatial resolution (*eg* Fig. 9a). Haeberli [4] used voronoi diagrams, placing the sites at random and filling each region with a colour sampled from the underlying image. This approach tesselates the image, but tiles all have different shapes, and do not attempt to follow edge features (Fig. 9b). Li and Milenkovic [9] have demonstrated an algorithm that places pattern pieces on cloth so that the least amount of cloth will be wasted when the pieces are cut out, effectively packing polygons densely. Their approach achieves packing efficiencies close to those of human experts, but the nature of the medium (cloth) dictates that the polygons must follow the grain of the cloth and not be rotated. Even so, packing arbitrary polygons into a rectangular container is an NP-hard problem [11]. The polygons in our problem are simpler, but must be rotated, so our goals are different. Moreover, we seek approximate, aesthetically pleasing packings; we do not need maximum-density packings.

Photomosaics [3, 12] approach the problem—representing an image with coarse tiles—by using spatial detail in the tiles themselves: each tile is an image, shrunk to size, on a rectangular grid. Their main task is searching a large database of images for one that approximates a block of pixels in the main image. Though impressive images are achieved in this way, their techniques do not address our needs. Another technique, “Escherization” [7], produces tilings of the plane using slightly-distorted versions of images, but relies on symmetry groups and regular tilings. We seek irregular dense tilings.

Work by Szeliski *et al* [13] resembles our approach. They use oriented particles to interpolate surfaces and to control surface deformations. Each surface particle exerts an orientation-dependent force on others. Particles are placed by finding low-energy configurations numerically. They accelerate the task of computing all inter-particle forces (the brute-force approach takes $O(n^2)$ time) through the use of a spatial hierarchy. Our approach uses graphics hardware to find analogous minimum-energy configurations. Deussen *et al* [1] use Lloyd’s method [10] to spread ink dots in a stippling pattern. Round ink dots have no preferred orientation, unlike square tiles, but otherwise their goals are very similar to ours. Hertzmann’s painterly renderings [5] emphasize edge features by using image gradients to orient progressively smaller curved paint strokes.

2.1 Challenges

To produce a smooth–looking flow of tiles, which follow the edges they want, mosaicists first set down tiles one at a time and then adjust all the tiles until the spacing is uniform. Only then do they cement the tiles into place. To reproduce this procedure algorithmically, we will also have to adjust each tile according to the direction field, and according to its neighbours’ positions. Each adjustment may require many other tiles to be adjusted in turn. To properly adjust tiles, an algorithm that simultaneously optimizes all tile positions is needed. We find such a method in the centroidal voronoi diagram.

2.2 Outline

The rest of this paper is organized as follows: Section 3 describes a generalization of centroidal voronoi diagrams, and a method that leverages graphics hardware to produce them. Section 4 presents a way to obtain the direction field ϕ from edge features in the image, while sections 5 and 6 address tile adjustments that may be used to enhance the final image. We present our results in section 7, and close with suggested future work in section 8.

3 Our Approach

Our approach positions tiles using the *centroidal voronoi diagram* (CVD). A voronoi diagram on the plane is defined by a collection of N sites, and divides the plane into N regions, such that all points within a region are closest to its associated site. Figure 4a shows an example. CVDs are voronoi diagrams with the additional property that each site is located at the mass-centre (centroid) of its region. Figure 4b shows a CVD.

CVDs are easily produced using Lloyd’s algorithm [10]: At each iteration, the algorithm moves each site to its region’s centroid, then re-computes the voronoi diagram. Its convergence properties are only known in one dimension [2], but it seems to work quickly in

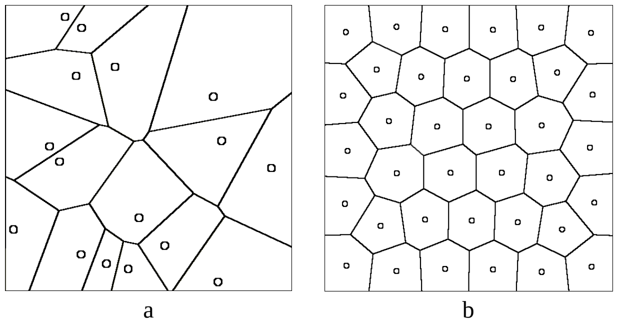


Figure 4: a) Voronoi diagram; b) Centroidal voronoi diagram.

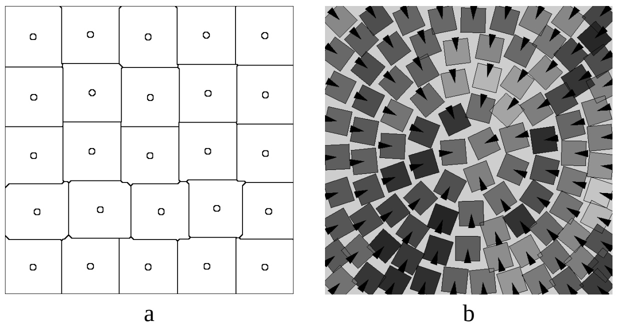


Figure 5: a) Lloyd's algorithm near convergence, for a manhattan metric; b) Tiles arranged on a curved direction field.

practice. As mentioned in [1], Lloyd's method suffers from fewer oscillations than other particle-spreading techniques.

CVDs are useful because they cover space fairly. One of their many uses includes sampling (they approximate a Poisson-disk point distribution), and they sometimes occur in nature (a honeycomb is a CVD). Locally, regions of a CVD often look like regular hexagonal tilings. This tiling property suggests adapting them to our purpose, but since we are using square tiles, we would prefer a CVD that produces locally *square*, not hexagonal tilings.

After some consideration, it becomes clear that CVDs tend to tile the plane with hexagons for the same reason that the densest packing of circles on the plane is hexagonal: such packings minimize the Euclidean distance $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. If we measure distance differently, we should get different optimal packings. In particular, square-grid packings minimize the manhattan distance metric $|x_1 - x_2| + |y_1 - y_2|$ (see Fig.5a). Hence CVDs for manhattan metrics will also be square tilings of the plane. If we adapt CVDs further by varying the metric's orientation, we should still obtain packings that locally look like square tilings (eg, Fig.5b). All that remains, then, is to find an efficient voronoi-diagram technique for variable-orientation manhattan metrics.

Hoff *et al* [6] present a method (first proposed by Haeberli[4]) that can be extended to do the job. To draw a voronoi diagram, they place an infinite cone at each site, with all apexes having the same z coordinate. They then render the cones using an orthogonal projection normal to the plane containing the sites. The rendering algorithm solves the visibility problem, which coincidentally also identifies the regions closest to each site. This is because a circular cone's explicit equation, $z = \sqrt{(x - x_0)^2 + (y - y_0)^2}$, embodies the Euclidean metric. If we use a square pyramid, $z = |x - x_0| + |y - y_0|$, we will obtain a voronoi diagram for the manhattan metric. The following algorithm then emerges:

Algorithm:

```
1.  $S \leftarrow$ list of random points on the image.
2.  repeat until converged:
3.          for each $p$ in S, place a square pyramid with apex at $p$.
4.          rotate each pyramid about the $z$ axis to align it
            with the direction field $\phi(p)$.
5.          render the pyramids with an orthogonal projection
            onto the $xy$ plane, producing a voronoi diagram.
6.          compute the centroid of each voronoi region.
7.          move each $p$ to the centroid of its voronoi region.
8.  draw a tile centred at each $p$, oriented along $\phi$.
```

The voronoi diagram in question can be approximated using the z-buffer algorithm available in most graphics cards. If each site's pyramid is drawn with a different colour, the corresponding voronoi regions will all have different colours. Note that this algorithm does not directly obtain the combinatorial version of the voronoi diagram—it does not identify each site's neighbours—but we do not need such information for our purposes. To be specific, step 6 of the algorithm is performed by reading back the frame buffer after drawing, and, for each cone colour, taking the average x and y of its pixels.

We expect this approach to produce reasonable tilings when the scale over which ϕ varies is much larger than the tile size. We have observed that if ϕ varies too rapidly, the algorithm will fail to converge. We leave a formal proof of this fact for future work. On the other hand, such small variations could never be captured by any tiling algorithm, since they fall under the minimum sample spacing available with fixed-size tiles.

4 Direction Field

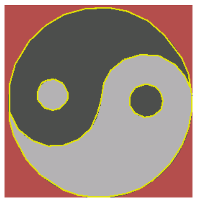
We now take up the issue of defining ϕ, the direction field that controls the orientations of the tiles. Since we will use tile directions to accentuate edges, directions should be based on edge features in the image. Generalized voronoi diagrams play a role here as well. The method is essentially the same as that described by Hoff [6].

The desired field $\phi(x, y)$ should follow an edge's orientation if (x, y) is near the edge. The gradient of the Euclidean distance from the edge provides such a field: If $s(t)$ is a planar curve (an edge feature), $P = (x, y)$ is a point not on the curve, and $C_s(x, y)$ is the point on s closest to P, then the function $d_s(x, y) = |P(x, y) - C_s(x, y)|$ is the point's minimum distance from the edge. If P is close to s, the line $\overline{C_s P}$ will be perpendicular to s.

For a collection S of curves, let $d_S(x, y)$ be the distance to the nearest curve in S. For each curve s in S, the surface whose explicit form is $z = d_s(x, y)$ resembles a long, curved mountain (s is the mountain ridge). If we 1) draw such a mountain for every curved edge feature in the image (with an orthogonal projection, again using the z-buffer algorithm), 2) read back the z-buffer, and 3) numerically evaluate the gradient ∇z at each pixel, we obtain an image-precision form of ϕ, the direction field we need. In Figure 6b we see perspective view of a set of curves, with associated ridges, and the derived direction field.

5 Tile Variations

Now that we have a method for placing tiles, we can apply variations to make the tilings more expressive. The algorithm's output is a set of points (the locations of the tiles). Each point has an associated orientation, which is applied to the tile. We can apply other attributes to the tiles, such as colour, size, aspect ratio, and shape.


a

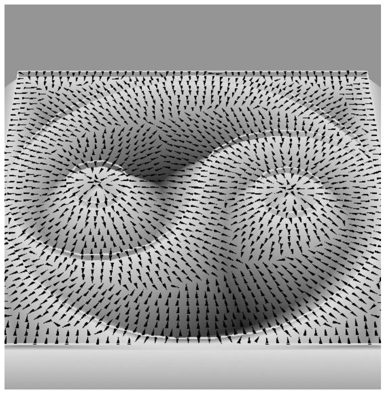

b

Figure 6: Calculating the direction field: a) The original image, with edge features drawn in yellow; b) The derived direction field.

5.1 Colour

To create a mosaic version of an image, tile colours should represent the image region they cover. Each tile colour may be either a point sample from the pixel at the tile's centre, or an average over the pixels covered by the tile. Point samples work best for images with uniformly-coloured regions, while area samples suit continuous-tone images.

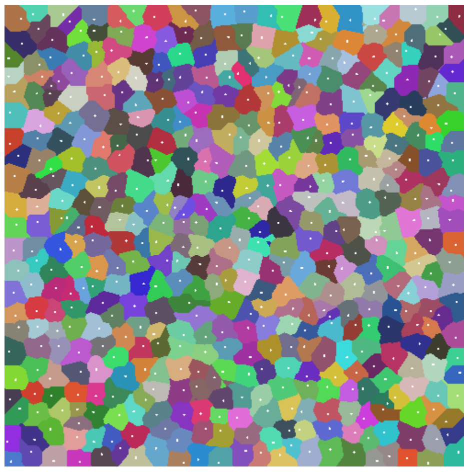
5.2 Size

Each tile site carries only position and orientation information, not tile size, but we can obtain a good tile size by equating the total tile area with the image size. For an $h \times w$ pixel image with n tiles, this yields tiles with sides of $d = \delta\sqrt{hw/n}$ pixels. The factor $\delta < 1$ accounts for packing inefficiencies due to variations in ϕ, although a value of $\delta = 0.8$ works in most cases, if all tiles are the same size.

As described above, variations in ϕ smaller than a tile width will not be captured by the algorithm. If such variations *must* be expressed, smaller tiles may be used locally. Smaller tiles may also be used in visually interesting areas, such as on lips and eyes in a portrait. The algorithm can be adapted to use varying tiles sizes, by adding a slope variation α in the cone equation $z = \alpha(|x - x_0| + |y - y_0|)$. If all tiles are positioned uniformly at first, and their sizes are adjusted according to their position in the image, Lloyd's method will eventually pack them more densely where smaller tiles are needed. However, sharp boundaries in tile size greatly slow the convergence. Good tilings can be obtained much faster by using rejection sampling to guide the initial sample positions. An initial tile position will be accepted with probability $(s_{min}/s)^2$, where s is the tile size and s_{min} is the smallest tile size in the image.

5.3 Aspect Ratio

Often, fine detail is needed only near a curve, or in other places where the direction field ϕ must be strongly emphasized. We can achieve this effect by using longer, narrower tiles where necessary. Such tiles effectively visualize the underlying direction field. Mosaicists use this effect to simulate strands of hair. The effect can be approximated by scaling the cone slopes non-uniformly, with a variation β in the cone equation: $z = \beta|x - x_0| + (1/\beta)|y - y_0|$. Again, this adjustment must be applied not only to the final tiles, but at each iteration, while regions are repositioned.


a

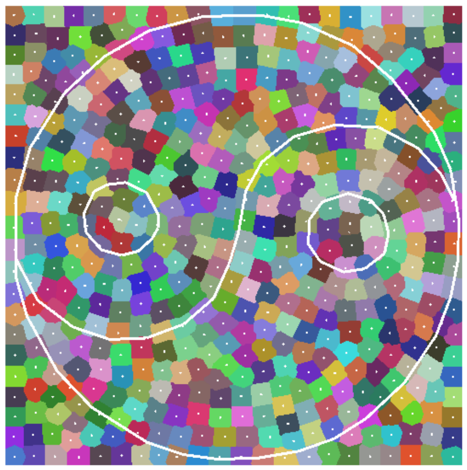
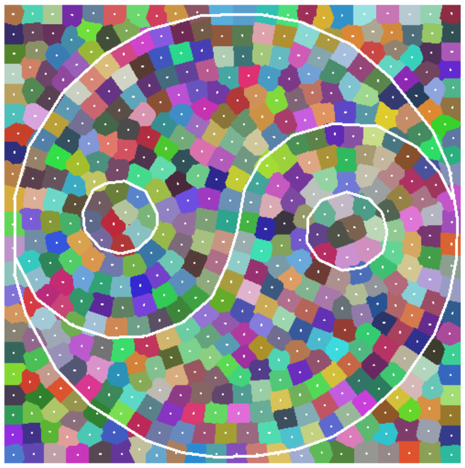

b

Figure 7: a) Initial voronoi diagram, with randomly placed tiles; b) Voronoi diagram after 20 iterations.

6 Edge Avoidance

On opposite sides of an edge, the direction of ϕ will change by $180°$. However, square tiles are radially symmetric, so this change does not come into play in Lloyd's algorithm, and is not reflected in the final result. Hence tiles near an edge will be oriented correctly, but nothing prevents them from straddling the edge. Thus the edge will lose definition. Fig. 1a illustrates this problem.

This problem can be easily overcome. At each iteration, each site is moved to its region's centroid. If some part of this region were to be overwritten with a different colour, the calculated centroid would change. By drawing the edges as thick lines with a distinct colour, the straddlers' centroids will be displaced away from the edge (Fig. 1b), and each iteration will propel the sites away from the edge (Fig. 1c). The withdrawal ceases when the tiles move off the edge (Fig. 1d). These sites in turn tend to push their neighbours away from the edge too, and the net result is to divide the mosaic into clearly defined regions with gaps where the edges once were (Fig 1e). A few iterations without the edges drawn will then fill in the gaps without spoiling the edge definition (Fig. 1f).


a

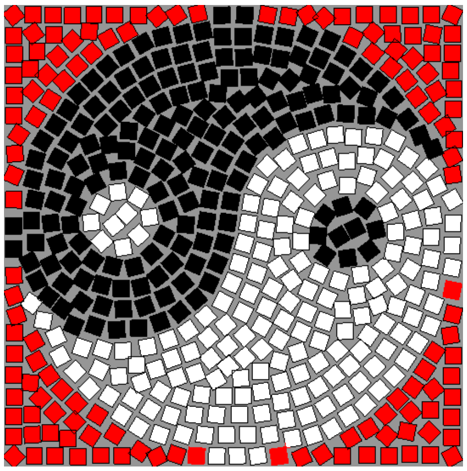

b

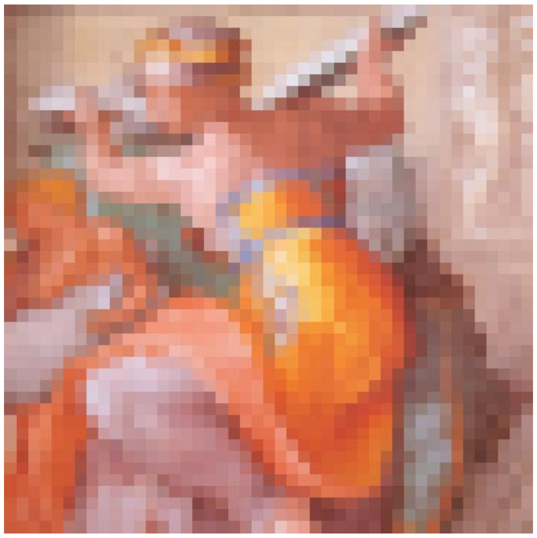
Figure 8: a)Voronoi diagram after edge avoidance; b) Final tiling, using point sampling for tile colours.

7 Results

The algorithm's progress is presented in Figures 6 through 8b. Fig. 6a shows the original image, with manually chosen curves superimposed where edge features must be emphasized, and Fig. 6b shows how ϕ is derived. The perspective view shows ridges at each edge, and the height gradient that defines ϕ. Fig. 7a shows initial tile sites, with their voronoi diagram, and Fig. 7b shows the final positions after 20 iterations. Notice that many tiles straddle edge features. The edge-avoidance technique described in the previous section is then applied, moving the voronoi regions off the edges (Fig. 8a), yielding the final image in Fig. 8b.

As we discussed in the introduction, a mosaic made up of N tiles conveys more information than an image made up of N pixels. We can see this in action by comparing Fig. 9a, an image of Michelangelo's *Lybian Sibyl* that uses 2025 pixels, with Fig. 10a, which better conveys the curving edges in the oracle's arms and robe, using the same number of tiles. The image can be improved further by varying tile sizes judiciously according to background, figure and detail. Background, figure and detail regions appear in blue, green and magenta, respectively in Fig. 10b. The figure also shows user-selected curves and the derived orientation field ϕ. The final mosaic appears in Fig. 10c.

Variable-sized tiles are again used for Hopper's *Second Story Sunlight*, in Fig. 11a. In Fig. 11b, based on a photograph of a stained-glass window, tiles are aligned along dark leading lines in the image. Both these tilings are based on images with clearly delineated colour regions.


a

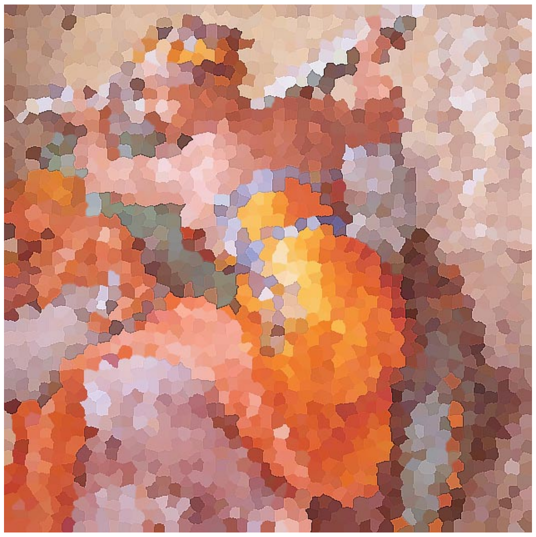

b

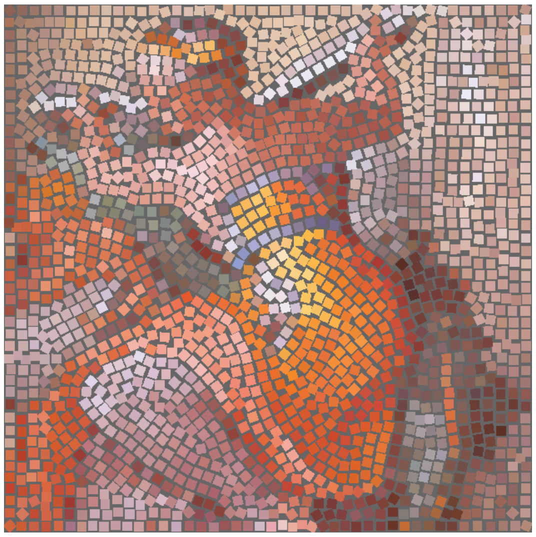
Figure 9: Lybian Sibyl a) using 2025 pixels; b) using Haeberli's technique.

Figure 11c uses elliptical tiles instead of rectangular ones. In this image, the tile size is chosen large enough to force overlaps between neighbours. Used in this way, the method serves to distribute "paint" strokes over the image, creating a painterly effect.

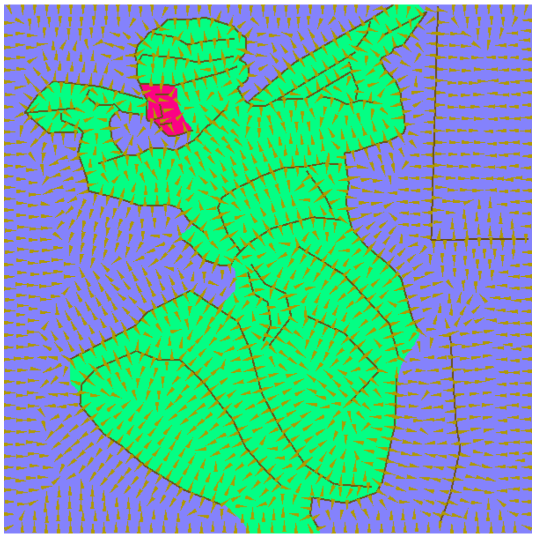
Fig. 12a uses elongated tiles to emphasize the vertiginous curved paint strokes in Munch's *The Scream*. The straight tile edges distract the eye from the curved paint strokes. Fig. 12b is more effective. It was created using Haeberli's method, filling in the voronoi regions corresponding to tiles in Fig. 12a.

Fig. 13 shows tilings based on photographs. We can see that the lower-contrast image of the cat fares more poorly than the seal because the figure's furry boundary is not well represented by a sharp change in tile size. Perhaps a gradation in tile sizes might be more effective.

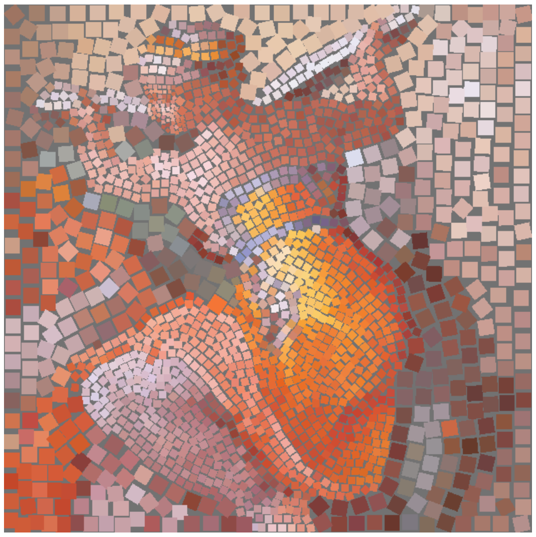
The algorithm is fast, though not real-time. Usually about 20 it-



a

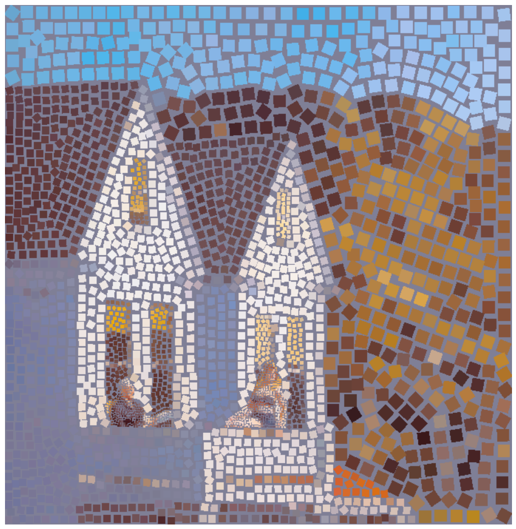


b

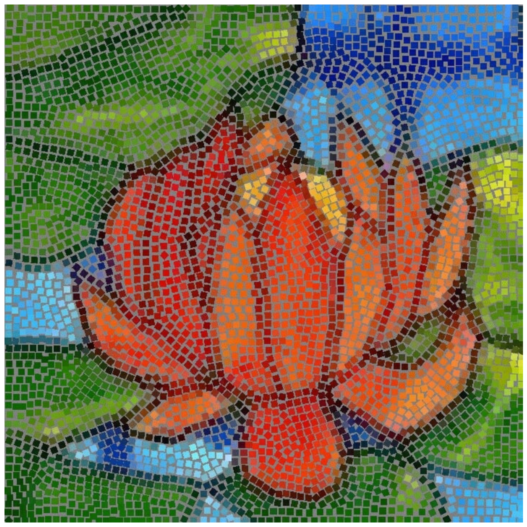


c

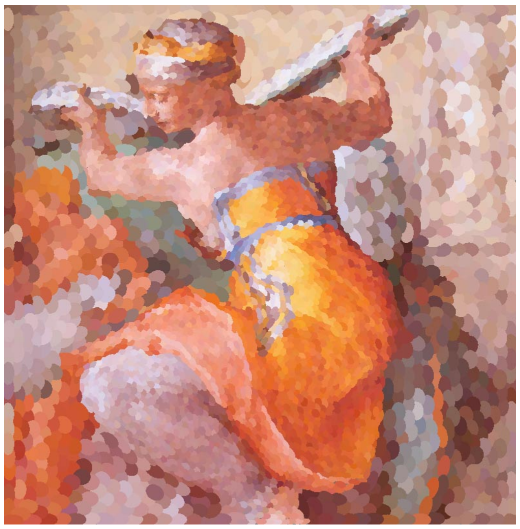
Figure 10: Lybian Sibyl a) using 2000 equal-sized tiles; b) background, figure and detail regions; c) using 2000 tiles in three sizes.



a

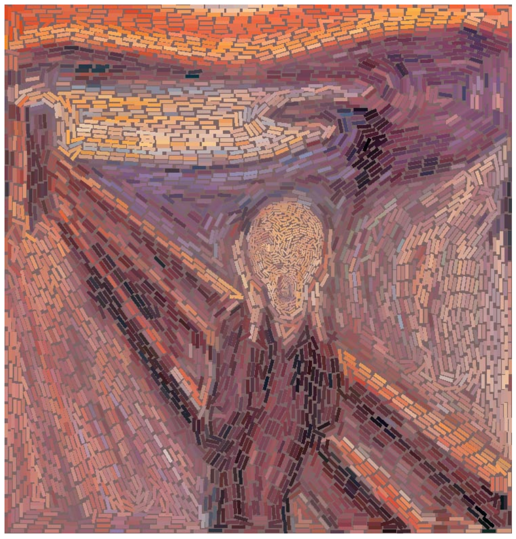


b

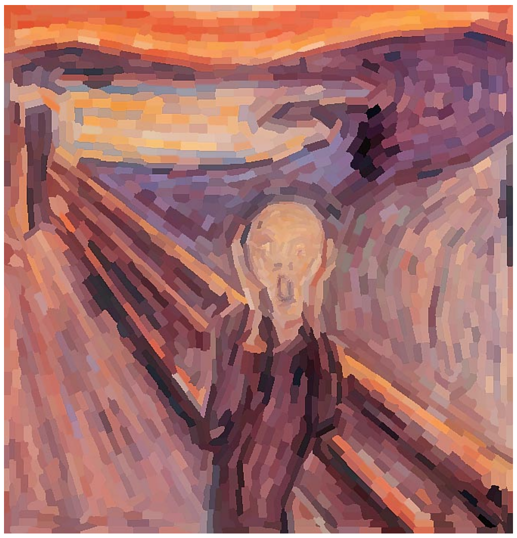


c

Figure 11: a) Hopper's "Second Story Sunlight"; b) Mosaic stained glass, with dark leading lines emphasized; c) Sibyl using overlapping oval tiles.

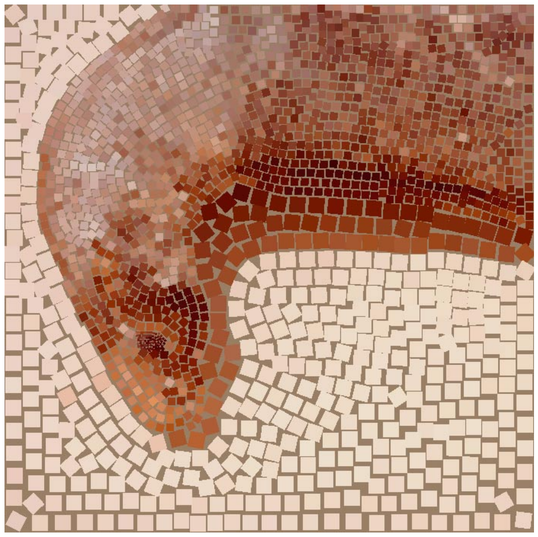


a

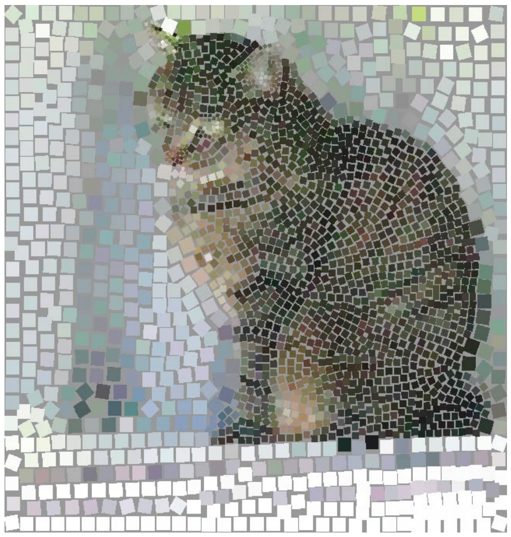


b

Figure 12: Munch's "The Scream", using a) long thin tiles and b) Haeberli's method on the corresponding voronoi regions.



a



b

Figure 13: Tiled photographs of a) seal on beach b) cat in window.

erations suffice for an image with 1000 tiles. Each iteration takes about one second for a 900×900 pixel image, on an 600 MHz Pentium III workstation, with an Nvidia 32 MB Geforce Plus graphics card. The time is dominated by the graphic card's I/O, which limits the frame-buffer readback needed to compute voronoi region centroids. The resulting tiling is visually satisfactory long before an exact CVD is obtained.

8 Conclusion

We have presented a new method to pack similar-shaped objects along an imposed direction field, and applied it to simulating decorative mosaics. The method produces good simulations, and is general enough to be applied to other problems where a minimum-energy configuration of particles is needed. The present work suggests avenues for further research.

8.1 Future Work

The current approach uses only the first-order moment of each voronoi region to obtain the region's centroid, but higher moments may yield more information. This may yield curvilinear, distorted tiles which better follow the direction field and are densely packed. Of course, this goes beyond the practices of traditional mosaicists.

Another way to obtain higher coverage is to "fill in" the gaps between tiles, which artisans do using small tile chips. Identifying the gaps requires neighbour information, which is not explicit in the image-precision approach used here, but which can be easily obtained by finding adjacent distinct pixels. Some slight improvement may be achieved by distorting square tiles into trapezoids, with distortion proportional to the direction field's divergence $\nabla \cdot \phi = \nabla^2 z$. For this to be effective, edges must be at least G_1 smooth.

The technique by which ϕ is derived is not the only one possible, and has the drawback of extending the edge's influence a long distance from it. More sophisticated user-defined direction fields may be used. The packing algorithm assumes no special properties in ϕ, so many choices are possible.

The method's use of initially random site positions is effective because all tiles have the same shape. Thus it cannot be directly used to pack polygons of different shapes. However, it may prove fruitful in the later stages of such packing algorithms, when small adjustments must be made.

Acknowledgments

I would like to thank Leslie Trueman for writing assistance, and the anonymous referees for many suggestions that improved the paper.

References

[1] Deussen O., Hiller, S., va Overveld, C. and Strothotte T. Floating Points: A Method for Computing Stipple Drawings. *Eurographics 00* 19:3.

[2] Du, Q., Faber, V. and Gunzburger, M. Centroidal Voronoi Tessellations: Applications and Algorithms. *SIAM Review 41* (1999): 637-676.

[3] Finkelstein, A. and Range, M., Image Mosaics, in Roger D. Hersch, Jacques André, and Heather Brown (eds.), *Electronic Publishing, Artistic Imaging and Digital Typography*, Proceedings of the EP'98 and RIDT'98 Conferences, St Malo: March 30 - April 3, 1998, Lecture Notes in Computer Science Series, number 1375, Heidelberg: Springer-Verlag 1998.

[4] Haeberli, P. Paint by Numbers. *SIGGRAPH '90* 207-214.

[5] Hertzmann, A. Painterly Rendering with Curved Brush Strokes of Multiple Sizes. *SIGGRAPH 98*: 453-460.

[6] Hoff, K., Keyser, J., Lin, M., Manocha, D. and Culver, T. Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware. *SIGGRAPH 99*: 277-286.

[7] Kaplan, C. and Salesin, D. Escherization. *SIGGRAPH 00*: 499-510.

[8] Hetherington, P. *Mosaics* London: Paul Hamlyn, 1967.

[9] Li, Z. and Milenkovic, V. Compaction and Separation Algorithms for Nonconvex Polygons and Their Applications. *European Journal of Operations Research* 84(1995): 539-561.

[10] Lloyd, S. Least Square Quantization in PCM. *IEEE Transactions on Information Theory* 28(1982): 129-137.

[11] Milenkovic, V. Rotational Polygon Containment and Minimum Enclosure. *Proceedings of the 14th Annual Symposium on Computational Geometry*, (June 1998): 1-8.

[12] Silvers, R. and Hawley, M. *Photomosaics*, New York: Henry Holt, 1997.

[13] Szeliski, R. and Tonnesen, D. Surface modeling with oriented particle systems. *SIGGRAPH 92*: 185-194.